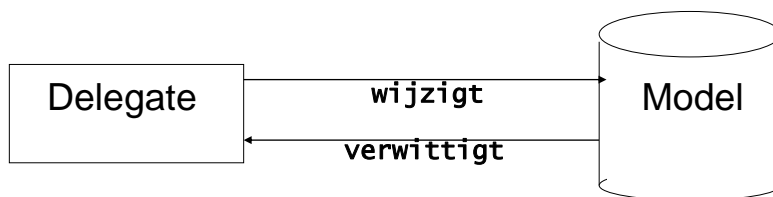


MODEL-VIEW-CONTROLLER:
DEEL 2
ListView

1

1. Delegate-Model architectuur

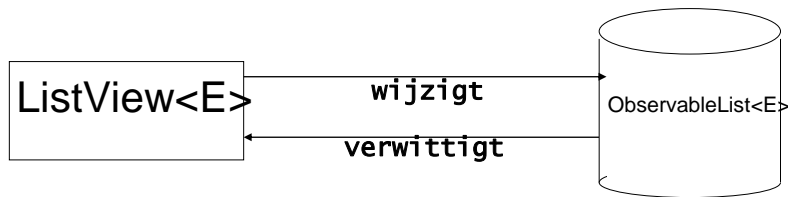


Delegate: View en controller worden samengebracht tot één enkel object, de delegate.

Model: de data

Het **delegate-model** wordt in javaFx-componenten toegepast, zoals **ListView**, TableView en TreeView.

2. ListView<E>



- **ListView<E>** voorziet alle functionaliteiten om data in een lijst te presenteren.
- **ListView<E>** ondersteunt de scheiding van data, presentatie en invoerverwerking, nl. de **delegate-model** architectuur.

3

2. ListView<E>

- De **observableList** verwittigt (**notifies**) de listView-component bij wijziging, toevoeging en verwijdering van de data.
- De interface **ObservableList<E>** (**het model**) voorziet methoden voor het opvragen van de data.
- De **listView** (**de delegate**) ondervraagt het model voor de opbouw van de lijst en zal het model wijzigen op basis van user input.

HoGent

2. ListView<E>

- Een listView bevat een **SelectionModel<E>**.

Een **selectionModel** weet welke elementen geselecteerd zijn uit de lijst. We kunnen instellen of er één of meerdere elementen kunnen geselecteerd worden.

- listView.**getSelectionModel()**.getSelectedItem() geeft het geselecteerde element terug.
- listView.**getSelectionModel()**.getSelectedItems() geeft de geselecteerde elementen terug.
- listView.**getSelectionModel()**.getSelectedIndex() geeft de index van het geselecteerde element terug.
- listView.**getSelectionModel()**.getSelectedIndices() geeft de indices terug van de geselecteerde elementen terug.

2. ListView<E>

- Slechts één element kan geselecteerd worden

```
listView.getSelectionModel().setSelectionMode(  
    SelectionMode.SINGLE);
```

- Defaultwaarde = SelectionMode.SINGLE

- Meerdere elementen kunnen geselecteerd worden

```
listView.getSelectionModel().setSelectionMode(  
    SelectionMode.MULTIPLE);
```

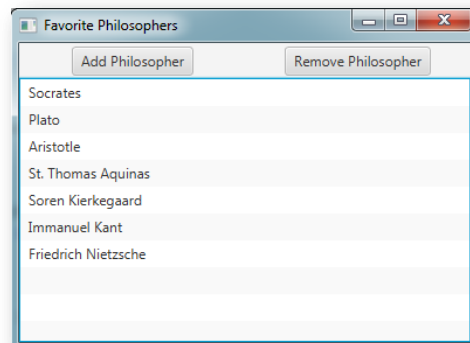
HoGent

3. ListView<E> & ObservableList<E>

- Interface ObservableList<E>
extends java.util.List<E>, Observable

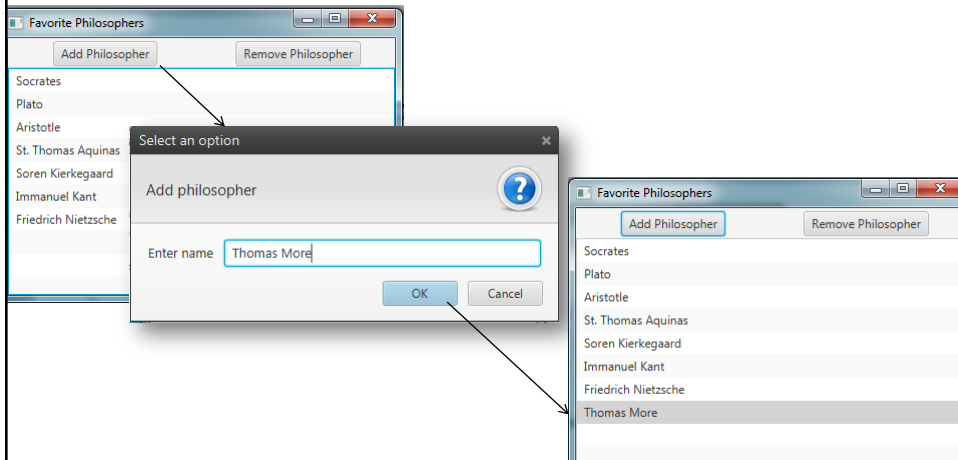
3. ListView<E> & ObservableList<E>

- Voorbeeld: een lijst met filosofen wordt weergegeven



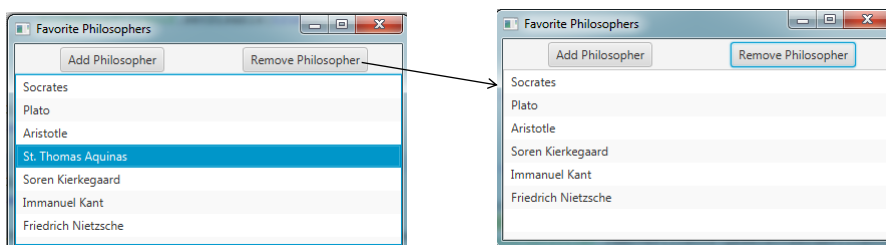
3. ListView<E> & ObservableList<E>

- De gebruiker kan filosofen toevoegen



3. ListView<E> & ObservableList<E>

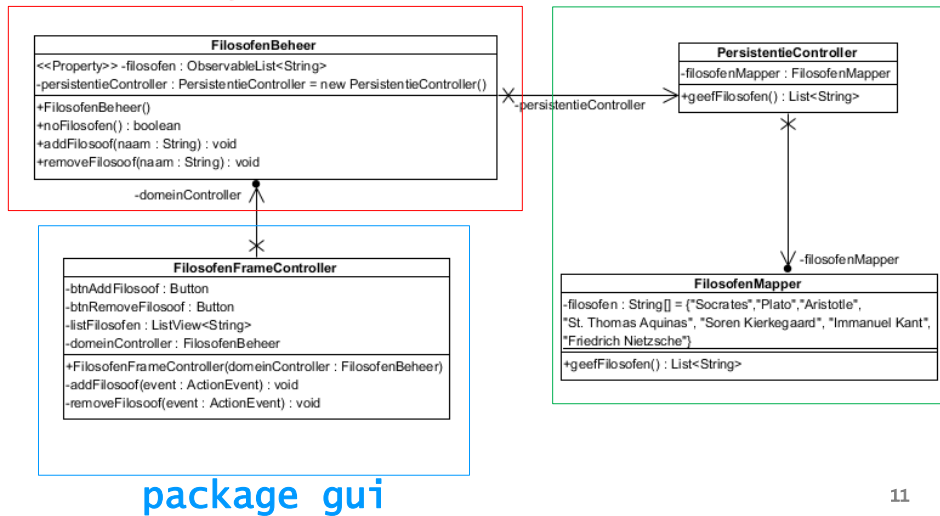
- En de gebruiker kan filosofen verwijderen



3. ListView<E> & ObservableList<E>

package domein

package
persistentie



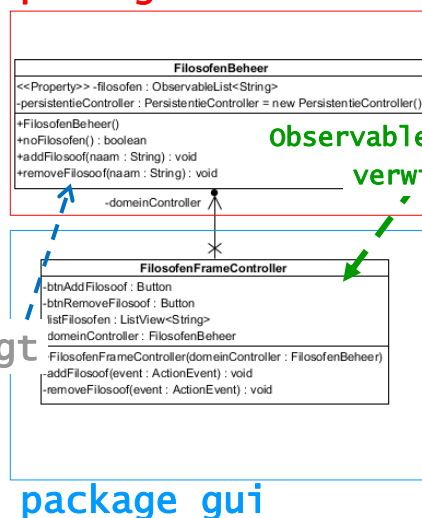
11

3. ListView<E> & ObservableList<E>

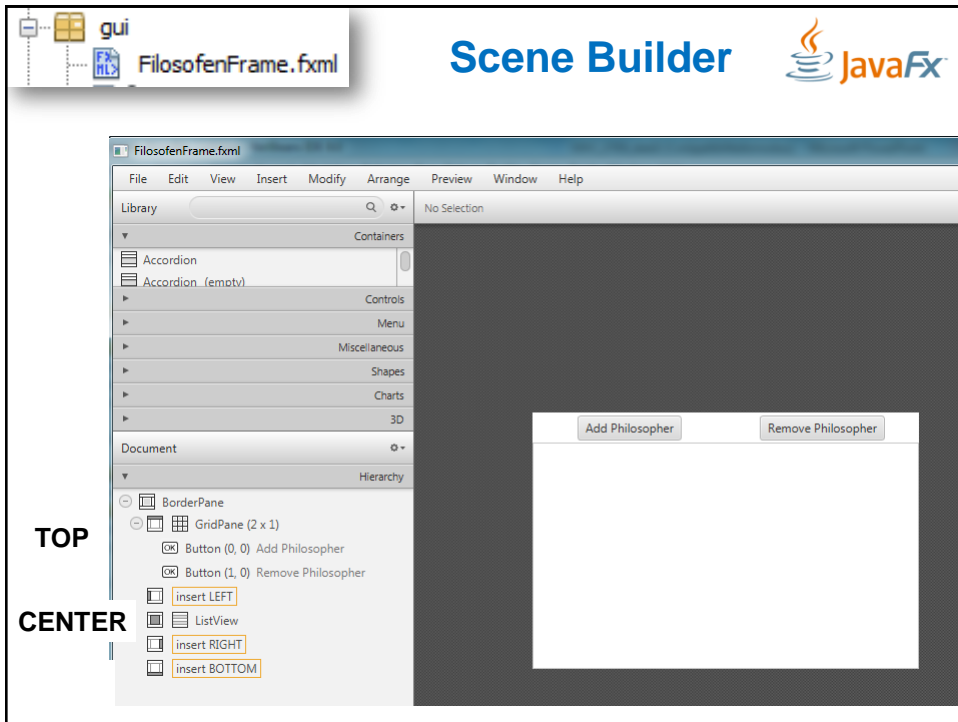
package domein

ObservableList<String>
verwittigt

ListView<String>
wijzigt



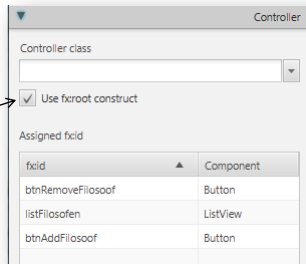
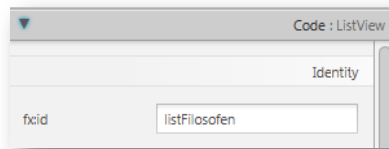
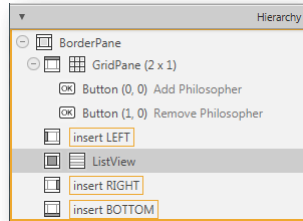
12



Scene Builder



ListView:



HoGent

```
package gui;
import ...;
```

```
public class FilosofenFrameController extends BorderPane {
```

```
    @FXML
```

```
    private Button btnAddFilosoof;
```

```
    @FXML
```

```
    private Button btnRemoveFilosoof;
```

```
    @FXML
```

```
    private ListView<String> listFilosofen;
```

```
    private FilosofenBeheer domeinController;
```

HoGent


```

public FilsofenFrameController(FilsofenBeheer
domeinController) {

    this.domeinController = domeinController;

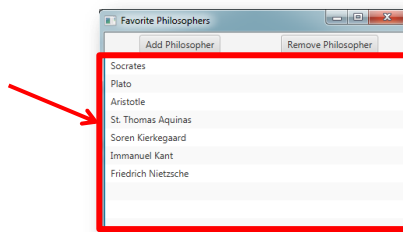
    FXMLLoader loader =
        new FXMLLoader(getClass().getResource(
            "FilsofenFrame.fxml"));

    loader.setRoot(this);
    loader.setController(this);
    try {
        loader.load();
    } catch (IOException ex) {
        throw new RuntimeException(ex);
    }
}

```

HoGent

// vul de ListView op met de elementen van de ObservableList
listFilsofen.setItems(domeinController.getFilsofen());



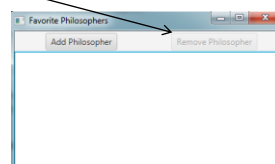
//Defaultwaarde = SelectionMode.SINGLE

```

domeinController.addObserver(e ->
btnRemoveFilsoof.setDisable(domeinController.noFilsofen()));
}
}

```

Indien
de lijst
leeg is



HoGent

@FXML

```
private void addFilosoof(ActionEvent event) {
```

// naam van filosoof aan de gebruiker vragen

```
    TextInputDialog dialog = new TextInputDialog();
```

```
    dialog.setTitle("Text Input Dialog");
```

```
    dialog.setHeaderText("Add philosopher");
```

```
    dialog.setContentText("Enter name:");
```

```
    dialog.showAndWait()
```

```
        .ifPresent(response -> {
```

```
            if (!response.isEmpty()) {
```

// voeg nieuwe filosoof toe in model

```
                domeinController.addFilosoof(response);
```

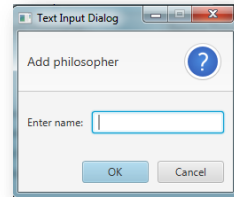
// zie volgende slide

```
                listFilosofen.getSelectionModel().selectLast();
```

```
            }
```

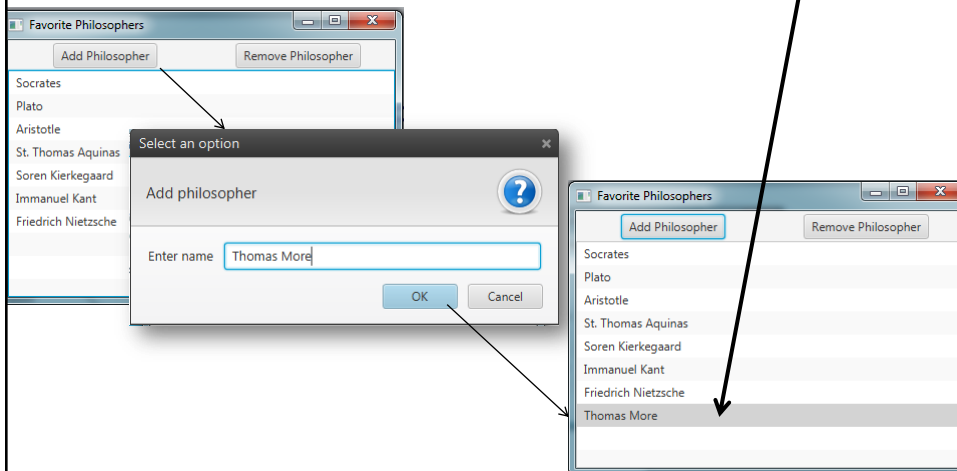
```
        });
```

```
    }
```



HoGent

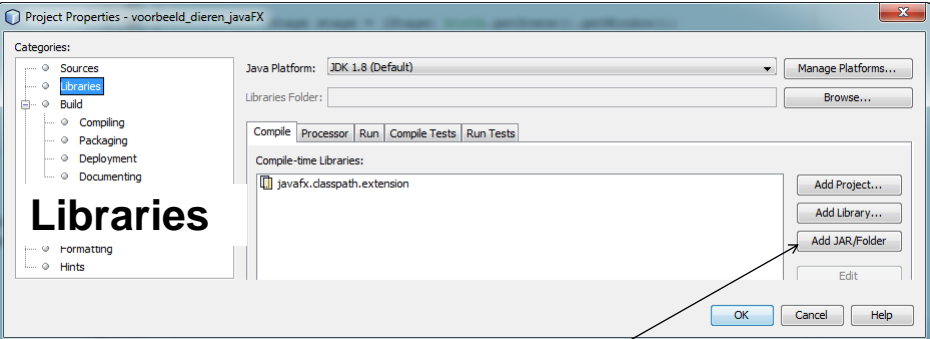
`listFilosofen.getSelectionModel().selectLast();`




20

Add Philosopher

Voorbeeld_filosofen_ListView → Properties



Add JAR/Folder
 openjfx-dialogs-1.0.2

of >= JDK 8u40

21

Remove Philosopher

@FXML

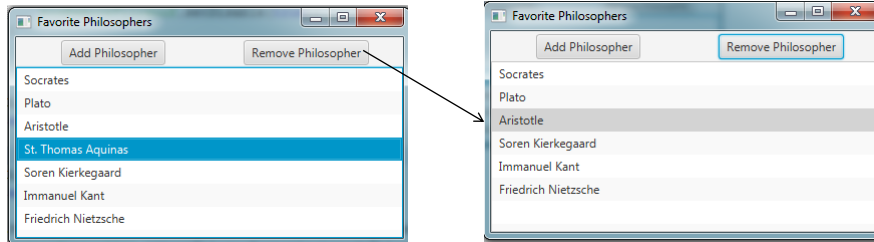
```
private void removeFilosoof(ActionEvent event) {

    // de geselecteerde filosoof opvragen
    String geselecteerdeFilosoof =
    listFilosofen.getSelectionModel().
        selectedItem();
    if (geselecteerdeFilosoof != null) {
        // zie volgende slide
        listFilosofen.getSelectionModel().clearSelection();

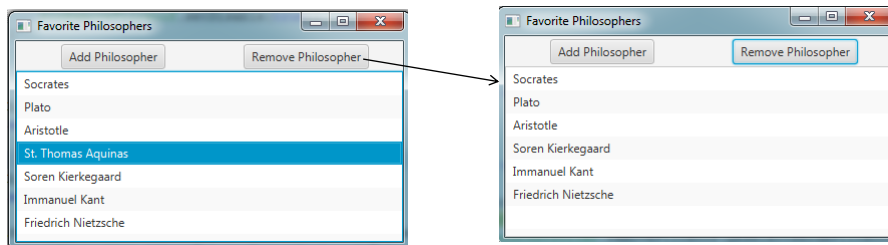
        // verwijder de geselecteerde filosoof in model
        domeinController.removeFilosoof(
            geselecteerdeFilosoof);
    }
}
```

HoGent

ZONDER `listFilosofen.getSelectionModel().clearSelection();`



MET `listFilosofen.getSelectionModel().clearSelection();`



`package domein;`

`ObservableList<E>`

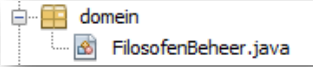
`import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import persistentie.FilosofenMapper;`

`public class FilosofenBeheer {`

`private ObservableList<String> filosofen;
private PersistentieController persistentieController =
new PersistentieController();`

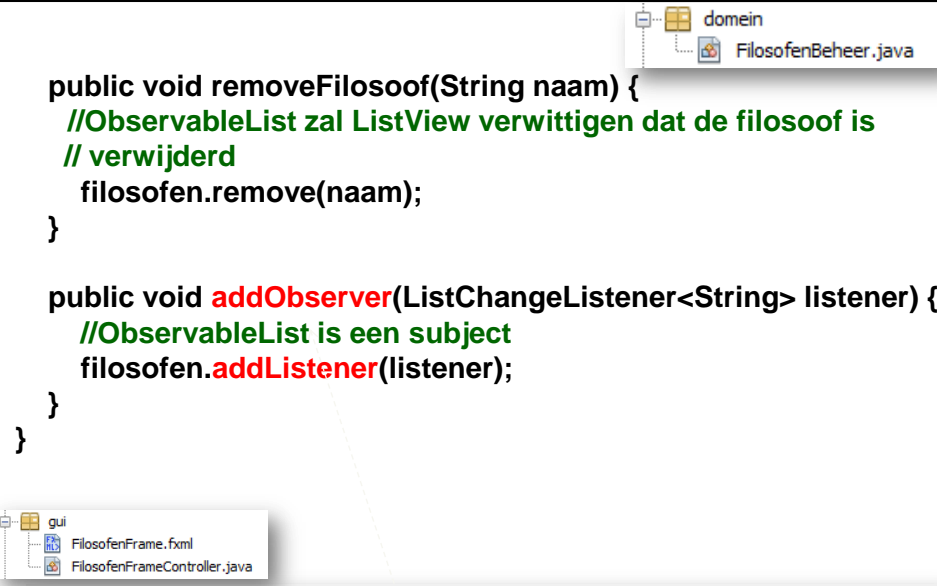
`public FilosofenBeheer() {
//creatie en opvullen van de ObservableList
filosofen = FXCollections.observableArrayList(
persistentieController.geefFilosofen()); //de
//persistentieController geeft een lijst van filosofen terug
}`

`public ObservableList<String> getFilosofen() {
return FXCollections.unmodifiableObservableList(filosofen); }
HoGent`

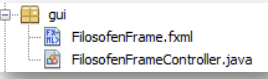


```
public boolean noFilosofen() {  
    return filosofen.isEmpty();  
}  
  
public void addFilosoof(String naam) {  
  
    if (naam != null && !naam.trim().isEmpty()) {  
  
        /*Een element wordt toegevoegd aan de ObservableList.  
        ObservableList zal ListView verwittigen (ingebouwde  
        Observer)  
        ListView zal de nieuwe filosoof weergeven */  
        filosofen.add(naam);  
  
    }  
}  
}
```

HoGent

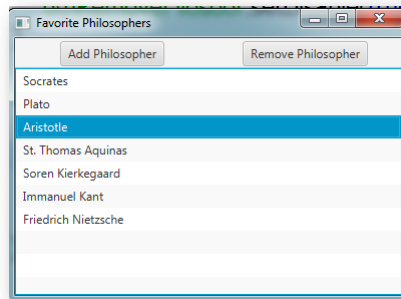


```
public void removeFilosoof(String naam) {  
    //ObservableList zal ListView verwittigen dat de filosoof is  
    // verwijderd  
    filosofen.remove(naam);  
}  
  
public void addObserver(ListChangeListener<String> listener) {  
    //ObservableList is een subject  
    filosofen.addListner(listener);  
}  
}
```



```
41 domeinController.addObserver(e  
42     -> btnRemoveFilosoof.setDisable(domeinController.noFilosofen()));
```

3.1 Rij selecteren



2 Aristotle

27

FilosofenFrameController

```
public FilosofenFrameController(FilosofenBeheer domeinController)
{
    ...
    listFilosofen.getSelectionModel().selectedItemProperty().
        addListener((observableValue, oldValue, newValue) ->
        {
            if (newValue != null) {
                int index = listFilosofen.getSelectionModel().getSelectedIndex();
                System.out.printf("%d %s\n", index, newValue);
            }
        });
}
```

28